Assignee: Intel Corporation
Docket No.: 2207/7942

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANTS	:	Deborah T. Marr, et al.
SERIAL NO.	:	09/490,172
FILED	:	January 22, 2000
FOR	:	ESTABLISHING THREAD PRIORITY IN A PROCESSOR OR THE LIKE
GROUP ART UNIT	:	2171
EXAMINER	:	Susan (Te Y.) Chen

M/S: APPEAL BRIEF - PATENT
COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

APPEAL BRIEF UNDER 37 CFR 41.37

Dear Sir:

This brief is in furtherance of the Notice of Appeal, filed in this case on July 6, 2006.

1. REAL PARTY IN INTEREST

Intel Corporation is the real party in interest for all issues related to this application.

2. RELATED APPEALS AND INTERFERENCES

There are no other appeals, interferences, or judicial proceedings known to Appellant or Appellant's legal representative, which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

3. STATUS OF THE CLAIMS

This application currently contains claims 1, 3-11, and 13-21. Claims 1, 3-11 and 13-21 are rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,105,127 to Kimura et al. in view of U.S. Patent No. 5,944,809 to Olarig et al.

4. STATUS OF AMENDMENTS

Applicants filed a Supplemental Amendment prior to the filing of this Brief in Response to the Final Office Action and Advisory Action to correct a minor antecedent basis issue. The attached listing of claims (section 8), reflects the status of the claims including this amendment.

5. SUMMARY OF THE INVENTION

Embodiments of the present invention pertain to establishing priority of a thread in a multi-threaded processor. A counter may be provided to store a value that assists in designating an amount of time that one thread will have priority over others. The value stored in the counter

may be selected for each thread so as to give some threads a greater amount of priority (e.g., in access to a resource) compared to others.

In the embodiment of claim 1, a method is provided including assigning a value in memory to indicate which of a plurality of threads executed by a single processor has a higher priority (see, e.g., pg. 6, lines 4-6 and elements 3 and 4 in Fig. 1). In a next operation, a resource is allocated between the plurality of threads depending on a priority assigned to each thread (see, e.g., pg. 8, lines 14-21; an example of a resource is shown in Fig. 4 (element 81) and pg. 9, lines 6-19). In the allocation, a counter is provided with a predetermined value for the plurality of threads (see, e.g., Fig. 3, element 68), the value being selected by control logic depending on the priority assigned to each thread (see, e.g., pg. 7, line 21 to pg. 8, line 15).

In the embodiment of claim 10 a method is provided including assigning a value in an APIC TPR (Advanced Programmable Interrupt Controller – Task Priority Register) register for a thread via execution of operating system code to indicate which of a plurality of threads executed by said single processor has a higher priority (see, e.g., elements 3 and 4 and pg. 6, lines 4-22). In a next operation a resource is allocated between the plurality of threads depending on a priority assigned to each thread (see, e.g., pg. 8, lines 14-21; an example of a resource is shown in Fig. 4 (element 81) and pg. 9, lines 6-19). In the allocation, a counter is provided with a predetermined value for said plurality of threads (see, e.g., Fig. 3, element 68), the value being selected by control logic depending on the priority assigned to each thread, said counter being used in said allocating operation (see, e.g., pg. 7, line 21 to pg. 8, line 15).

In the embodiment of claim 11, an apparatus is provided comprising a memory to store a value to indicate which of a plurality of threads to be executed by a single processor has a higher

priority (see, e.g., pg. 6, lines 4-6 and elements 3 and 4 in Fig. 1). A resource is allocated between the plurality of threads depending on a priority assigned to each thread in said memory (see, e.g., pg. 8, lines 14-21; an example of a resource is shown in Fig. 4 (element 81) and pg. 9, lines 6-19). A counter loaded with a predetermined value by control logic for each thread in the memory depending on the priority assigned (see, e.g., Fig. 3, element 68). The counter is used to allocate said resource between said plurality of threads (see, e.g., pg. 7, line 21 to pg. 8, line 15).

In the embodiment of claim 20, an apparatus is provided for establishing thread priority in a single processor. The apparatus includes an APIC TPR register for a thread wherein execution of operating system code causes a value to be stored in said register to indicate which of a plurality of threads to be executed by the single processor has a higher priority (see, e.g., elements 3 and 4 and pg. 6, lines 4-22). A resource allocated between said plurality of threads depending on a priority assigned to each thread in said memory (see, e.g., pg. 8, lines 14-21; an example of a resource is shown in Fig. 4 (element 81) and pg. 9, lines 6-19). A counter loaded with a predetermined value by control logic for each thread in said memory (see, e.g., Fig. 3, element 68), said value being selected depending on the priority assigned, said counter being used to allocate said resource between said plurality of threads (see, e.g., pg. 7, line 21 to pg. 8, line 15).

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. The rejection of claims 1, 3-11 and 13-20 under 35 U.S.C. § 103(a) as being unpatentable over Kimura et al., U.S. Patent No. 6,105,127 (hereinafter "Kimura") in view of Olarig et al., U.S. Patent No. 5,944,809 (hereinafter "Olarig").

7. ARGUMENT

Under 35 U.S.C. §102(b), a claim is invalid if the invention claimed therein is described in a patent issuing more than one year prior to the filing of the subject patent application. Though a patent reference may have issued early enough (or filed early enough as the case for 35 U.S.C. §102(e)), that reference must also enable one skilled in the art to practice the claimed invention. *See* Akzo N.V. v. U.S. Int'l Trade Comm'n, 1 U.S.P.Q.2d (BNA) 1241, 1245 (Fed. Cir. 1986).

Absent anticipation it may be possible to combine two or more patents together to render a claimed invention obvious, and unpatentable, under 35 U.S.C. § 103(a). In determining whether the claims are unpatentable it is necessary to look to what the references actually teach. "It is impermissible within the framework of § 103 to pick and choose from any one reference only so much of it as will support a given position, to the exclusion of other parts necessary to the full appreciation of what such reference fairly suggests to one of ordinary skill in the art." In Re Wesslau, 147 U.S.P.Q. (BNA) 391, 393 (C.C.P.A. 1965). Accordingly, a prior art reference must be considered in its entirety, and portions thereof must be taken in proper context. MPEP § 2141.02; Bausch & Lomb, Inc. v. Barnes-Hind, Inc., 230 U.S.P.Q. (BNA) 416, 419 (Fed. Cir. 1986).

Claims 1, 10, 11, and 20

As recited in independent claims 1, 10, 11, and 20, a counter is provided that has a value selected depending on the priority assigned to the thread. Allocating a resource between a plurality of threads is controlled based on the counter (see, e.g., pages 7-8 of the filed

application). In one embodiment, the counter value can bet set high when a high priority is given to a thread, and set to a low value when a low priority is given to a thread. The cited references fail to teach or suggest this feature of the independent claims.

As stated in the Final Office Action, the Examiner concedes that Kimura fails to disclose "a counter loaded with a predetermined value by control logic for each thread in the memory, such that the value being selected depending on the priority assigned, and the counter being used to allocate said resource between a plurality of threads." (Final Office Action, p. 3).

The Final Office Action relies on Olarig to make up for the deficiencies of Kimura. In particular, the Final Office Action points to LOPIC and COPIC as being "threads" and a two-bit counter used to allocate resources between LOPIC and COPIC. As shown in Fig. 3, multiple CPUs are shown (elements 15 and 106), each with a "LOPIC" (elements 305, 306, respectively; "local programmable interrupt controller). The Final Office Action first cites to Col. 3, lines 22-38. The first sentence of this paragraph states that it applies to "balancing the interrupt loading among various processors in a scalable MP [multiprocessor] system." A counter is associated with each processing unit, and that counter is incremented each time an I/O interrupt is dispatched to that processing unit (Col. 3, lines 36-38). In the prior art example of Fig. 1, four processors are provided (CPU1 105 being one of them), and a two-bit counter is associated with each CPU. Each CPU is initialized with a value 00, 01, 10, or 11 so that each processor has a different two bit value. When an I/O interrupt is sent to one of the processors, each of the processors' two-bit counters are incremented by 1. (Col. 6, lines 42-57). The counters serve to provide a "round-robin" system for allocating the interrupt to the appropriate processor. In effect, one of the counters will have the lowest (or highest) value, and each time an I/O interrupt

is sent to one of the processors, the counters are all incremented so that a new one of the processors will have the lowest (or highest) value. This assignment of the lowest (or highest) value is shared evenly among the four processors.

Claim 1, for example, provides a method where a single processor is to execute a plurality of threads. A counter is provided "with a predetermined value for [the] plurality of threads, [the] value being selected by control logic depending on the priority assigned to each thread." Neither of these features are shown in Olarig. First, Olarig is quite clearly associated with balancing workload for interrupts among several different processors, not a single processor executing such code. Second, Olarig does not provide control logic that selects the value for the counter "depending on the priority assigned to each thread." In Olarig, the values for the counters are essentially random in that it does not matter which processor has which value, just as long as the four values are different (avoiding any "ties" when seeking to have one of the processors execute code to handle an I/O interrupt). Thus, the counter in Olarig is merely a round robin identifier for each of the four processors to execute its own LOPIC code.

Though counters are well known in the art, there is no description or suggestion in the Kimura and/or Olarig references to provide a counter for allocating resources in a multi-threaded environment as described in claim 1. The remaining independent claims, claims 10, 11, and 20, include similar limitations as found in claim 1. Since features of the independent claims are neither shown nor suggested by the Kumra and/or Olarig references, the rejection of these claims and the claims that depend from them is in error. Accordingly, Appellants respectfully request that the rejection of claims 1, 3-11 and 13-20 under 35 U.S.C. § 103(a) be reversed.

CONCLUSION

Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's decision rejecting claims 1, 3-11, and 13-21 under 35 U.S.C. § 103(a) direct the Examiner to pass the case to issue.

The Commissioner is hereby authorized to charge the appeal brief fee of $500.00 and any additional fees which may be necessary for consideration of this paper to Kenyon & Kenyon Deposit Account No. 11-0600. A copy of this sheet is enclosed for that purpose.

Respectfully submitted,

Date: January 16, 2007

Shawn W. O'Dowd
(Reg. # 34,687)

KENYON & KENYON LLP
1500 K Street, NW
Suite 700
Washington, DC 20005
(202) 220-4200 telephone
(202) 220-42501 facsimile
DC1-643267

APPENDIX

(Brief of Appellants D. Marr et al.
U.S. Patent Application Serial No. 09/490,172)

8. CLAIMS ON APPEAL

The claims in their current form (including those claims under appeal) are presented below:

1. A method comprising:

assigning a value in memory to indicate which of a plurality of threads executed by a single processor has a higher priority;

allocating a resource between said plurality of threads depending on a priority assigned to each thread; and

providing a counter with a predetermined value for said plurality of threads, said value being selected by control logic depending on the priority assigned to each thread, said counter being used in said allocating operation.

2. (Canceled)

3. The method of claim 1 wherein in said allocating step, a first thread is given greater access to the resource than other threads when said first thread is assigned a higher priority than said other threads.

4. The method of claim 1, wherein in said allocating step, the other threads are given greater access to the resource than the first thread when said first thread is assigned a higher priority than the other threads and is not using said resource.

5. The method of claim 3 wherein said resource is a unit in a processor system.

6. The method of claim 5 wherein said resource is a decode unit.

7. The method of claim 6 further comprising:

providing instructions from a first thread to a first queue;

providing instructions from a second thread to a second queue;

supplying a first number of instructions to said decode unit from said first queue;

supplying a second number of instructions to said decode unit from said second queue;

selecting said first and second numbers based on said value in memory.

8. The method of claim 3 wherein said resource is a bus.

9. The method of claim 8 further comprising:

providing bus requests from a first thread to a first queue;

providing bus requests from a second thread to a second queue;

servicing a first number of bus requests from the first queue;

servicing a second number of bus requests from said second queue; and

selecting said first and second numbers based on said value in memory.

10. A method comprising:

assigning a value in an APIC TPR register for a thread via execution of operating system code to indicate which of a plurality of threads executed by a single processor has a higher priority;

allocating a resource between said plurality of threads depending on a priority assigned to each thread; and

providing a counter with a predetermined value for said plurality of threads, said value being selected by control logic depending on the priority assigned to each thread, said counter being used in said allocating operation.

11. An apparatus comprising:

a memory to store a value to indicate which of a plurality of threads to be executed by a single processor has a higher priority;

a resource allocated between said plurality of threads depending on a priority assigned to each thread in said memory; and

a counter loaded with a predetermined value by control logic for each thread in said memory depending on the priority assigned, said counter being used to allocate said resource between said plurality of threads.

12. (Canceled)

13. The apparatus of claim 11 wherein a first thread is given greater access to the resource than other threads when said first thread is assigned a higher priority than said other threads.

14. The apparatus of claim 11 wherein the other threads are given greater access to the resource than the first thread when said first thread is assigned a higher priority than the other threads and is not using said resource.

15. The apparatus of claim 13 wherein said resource is a unit in a processor system.

16. The apparatus of claim 15 wherein said resource is a decode unit.

17. The apparatus of claim 16 further comprising:

a first queue to store instructions from a first thread;

a second queue to store instructions from a second thread;

control logic coupled to said first and second queues and said decode unit, said control logic to permit a first number of instructions to be transferred to said decode unit then a second number of instructions to be transferred to said decode unit, said first and second numbers being selected based on said value in memory.

18. The apparatus of claim 13 wherein said resource is a bus.

19. The apparatus of claim 18 further comprising:

a bus unit including

a first queue storing bus requests from a first thread;

a second queue storing bus requests from a second thread;

control logic coupled to said first and second queues, said control logic to control servicing of a first number of bus requests from the first queue and a second number of bus requests from said second queue, said first and second number being selected based on said value in memory.

20. An apparatus for establishing thread priority in a single processor comprising:

an APIC TPR register for a thread wherein execution of operating system code causes a value to be stored in said register to indicate which of a plurality of threads to be executed by said single processor has a higher priority;

a resource allocated between said plurality of threads depending on a priority assigned to each thread in said memory; and

a counter loaded with a predetermined value by control logic for each thread in said memory, said value being selected depending on the priority assigned, said counter being used to allocate said resource between said plurality of threads.

21. The apparatus of claim 20 wherein a first thread is given greater access to the resource than other threads when said first thread is assigned a higher priority than said other threads.

9. EVIDENCE APPENDIX

No further evidence has been submitted with this Appeal Brief.

10. RELATED PROCEEDINGS APPENDIX

Per Section 2 above, there are no related proceedings to the present Appeal.